Using information centric networking for mobile devices cooperation at the network edge

Using Information Centric Networking for Mobile Devices Cooperation at the Network Edge

Fabio Malabocchia[*], Romeo Corgiolu[*], Maurizio Martina[†], Andrea Detti[‡], Bruno Ricci[‡], Nicola Blefari-Melazzi[‡]
[*]Telecom Italia - Turin, Italy
[†]Department of Electronics, Politecnico di Torino, Turin, Italy
[‡]CNIT - Department of Electronic Engineering, University of Rome "Tor Vergata", Rome, Italy
Email: fabio.malabocchia@telecomitalia.it, romeo.corgiolu@gmail.com, maurizio.martina@polito.it,
{andrea.detti, bruno.ricci, blefari}@uniroma2.it

***Abstract*—This paper presents an Android application exploiting a collaborative multi-RAT approach (cellular and Wi-Fi Radio Access Technologies) for adaptively streaming video content. The application has been developed and tested on real Android devices of different vendors. Each device coordinates with the others to download portions of a video stream through its own preferred RAT, and shares the results with the neighboring devices through a proximity, high bit rate network based on Wi-Fi Direct. As a result, the aggregate bandwidth available to the group of collaborating devices allows a faster download and a higher quality of video playback streaming, with respect to what is achievable by a single device. The application is designed to exploit common Android devices along with the key functionalities of an Information Centric Network: routing-by-name, in-network caching and multicasting.**

Keywords—Information Centric Networking; live adaptive video streaming; cellular networks; MPEG DASH; test-bed, prototyping, mobile network applications.

I. INTRODUCTION

Video is rapidly becoming the primary source of traffic. According to [14], in 2018, the sum of all forms of IP video will be about 80-90% of all the traffic delivered to and from the end users. This video will be consumed on a number of different platforms spanning from smartphones to HD smart TVs. Content downloaded by a smartphone can be viewed on a TV through one of the many possibilities that are provided by new consumer electronic standards (e.g. DLNA, MHL/HDMI) or proprietary protocols like AllShare Play (Samsung) or Air Play (Apple).

Mobility is often traded off with resolution, since there is a direct correlation between the size of the screen and the traffic volume consumed. At the same time content can be accessed through a variety of access technologies, and some of them may not be optimal for the purpose (like capacity capped connections, or slow technologies such as EDGE networks).

In this scenario, common off-the-shelf devices are capable of accessing multiple Radio Access Networks at the same time, also using Device-to-Device technologies such as Wi-Fi Direct [6]. Application libraries and protocols like Alljoyn by Qualcomm and the Common Connectivity Framework by Intel allow developers to exploit locality in their applications.

This paper proposes an Information-Centric-Networking (ICN) enabled Android application for live adaptive video streaming in cellular networks. A group of mobile, off-the-shelf, un-rooted Android devices, cooperatively use their preferred RAT connections to achieve a better playback quality and offload the cellular network. In our application, each device coordinates with the other peers and downloads a portion of the stream and offers it locally to the other members of the group through a Wi-Fi Direct channel. Each device is thus able to recombine and play the whole video stream. Starting from the proof-of-concept of [7], we developed the application using the ICN architecture named Content Centric Network (CCN) [1], also known as Named Data Network (NDN), and evaluated the effectiveness of our solution using real Android hardware connected to real cellular networks. We show that it is possible to establish a collaborative network that exploits all the Radio Access Technologies (RAT) available on a device to maximize the quality of multimedia consumption.

II. BACKGROUND

A. Information Centric Networking

The concept of Information Centric Networking [1] [2] has been proposed to supersede the current paradigm of host-to-host communication. Among the various architectures proposed in the context of ICN ([1], [4], [5]), we chose the Content-Centric Network (CCN) architecture [1], which was implemented in the CCNx software. In a CCN, *contents* are addressed using unique hierarchical *names*. When a client wants to download a content, it will issue an "Interest" message, including the content name among its parameters. All interest messages are then *routed-by-name* by the CCN nodes, using a name-based Forwarding Information Base (FIB). When the "interest" reaches the intended content, a *Data* message is issued and transferred back on the same path of the request. In order to keep track of this reverse path, each node traversed by an interest records a <content, previous hops> tuple on a local *Pending Information Table* (PIT), which will then be used to forward the content to the requesting nodes.

The benefit of using CCNx for applications development is the fact that a lot of functionalities that may be hard and long to implement using the classical TCP paradigm (such as caching, routing by-name and multicasting) are offered out-of-box, thus any implementation of a networking application can take advantage of these functionalities without added effort.

B. Video standard

We used the MPEG-DASH (Dynamic Adaptive Streaming over HTTP) scheme [3], in which each video is represented by

a manifest file (*MPD file*), which contains some meta-information of the video to be streamed (codec, base URL and so on), and by a series of video *segments* (*M4S files*), each of which represents a portion of the video at different bitrates and is identified in the manifest file with a unique URL. The main motivations for choosing DASH are its popularity and the possibility to dynamically adapt the bitrate of the stream on the changing radio propagation conditions experienced by each peer on its preferred RAT and within the group of peers.

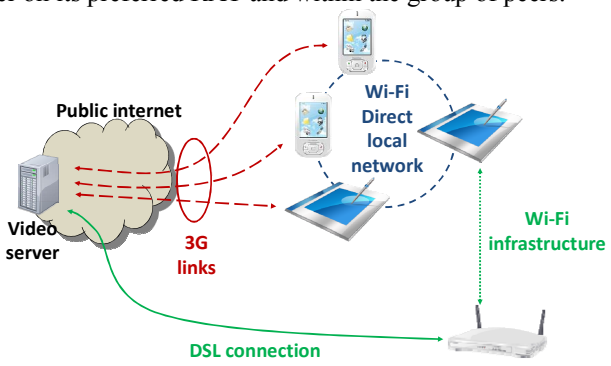

Fig. 1: structure of a video peer

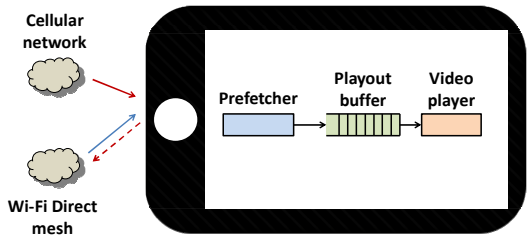

Fig. 2: structure of a video peer

C. *Previous works*

The application of ICN to the case of video streaming is not a novel topic, as there are in literature other solutions which deal with slightly different problems or objectives.

This paper is an evolution of [7], in which a proof-of-concept application for the same purpose was developed using a Java implementation running on plain laptops, without dealing with the possible limitation of a real off-the-shelf mobile device. In this paper we ported the proof-of-concept application on commercial Android devices, thus facing all the limitations and problems that this environment and platform implies. In addition, we report here a new extensive experimental campaign carried out on real Android devices.

Similar problems are treated for example in [8] (streaming of single rate on demand videos), [10] (TCP/IP streaming of on demand single rate videos) and [9] (streaming using fixed network and client server approach). Streaming on fixed network is also treated in [13].

The concept of using multiple access networks to get the same data has been used also in [12]. Nevertheless, the solution proposed is based on using just *one* of the possible available access networks, while in our paper we use *all* possible cellular links, thus avoiding the limitation of the most performing *single* gateway.

This solution presented in our paper may also be similar to the problem of "multi-homed" video streaming [11]: the main difference is that in that case all the different links used to get the data are hosted by the same device, and not by multiple devices.

A similar concept of our application is done in [16], but while the authors have used rooted devices, our approach consider plain non-modified Android devices.

III. THE APPLICATION

In this section we review the logic of our application [7].

A. *Scenario*

We consider a scenario (Fig. 1) in which some nearby mobile devices are interested in watching the same video content from the public Internet. The use case we focused on is mainly stationary, when people watch multimedia streams standing still (for instance, a family in a private house that wants to see a certain movie from an internet movie provider). The public Internet can be reached by each mobile device using all the different *radio access networks* available in the device (cellular, Wi-Fi infrastructure, and so on). All the devices take charge of a small portion of the video stream from the public Internet, and share it with the other nodes using a local one-hop full mesh Wi-Fi Direct network. This is done in order to (a) reduce the cellular traffic on each mobile video peer and (b) achieve a better *global* video stream quality. Each device can reach the public Internet with the best link available, provided that all the available links may have different performance, different radio chains (LTE cat 4, DC-HSDPA, HSPA+, and so on), and may experience different coverage conditions and/or being subject to different data plan conditions. Of course WLANs can be used, but in case they are not available a devices may use the cellular network to reach the public Internet.

As stated in the introduction, the devices we consider are off-the-shelf un-rooted Android devices. Indeed, rooting a device means changing the OEM configuration and give users root permissions (super user capabilities), via a process that is device-dependent and tend to be reserved to users who have a great technical knowledge of the device. In addition, when not explicitly illegal, the rooting process voids the warranty of the device. Thus, using un-rooted devices allows to target mass market devices without breaking OEM or operator policies.

B. *Video repository*

The MPEG DASH video stream is kept in a CCNx repository available on the public Internet. The MPD is made available from the beginning of the live stream, while each M4S file is created and published on the fly. Since each peer can join the video stream anytime, there is a need for a mild synchronization between the nodes and the video source, to ensure that each video peer is playing (and thus, downloading) the same portion of the stream. To do so, when publishing a new segment, the video server also publishes a Video Timing Information (VTI), containing a reference to the last published video segment, along with the repository clock. This data unit is the first to be downloaded by each peer, in order synchronize with the video source and fetch the correct video segment.

The naming scheme used for all the video data units involved resembles the scheme used in [9]:

MPD ccnx:/server-prefix/video.mpd

M4S ccnx:/server-prefix/video_BW/video_SN.m4s

VTI ccnx:/server-prefix/video.vti

SN is the video segment number and BW is the coding rate. An example is 'ccnx:/foo.eu/video1_100/video1_3.m4s', in which the segment 3 of the video *video1* is found at the server prefix *foo.eu* and encoded at *100* Kbps.

C. Video peer

A video peer (Fig. 2) is composed of three *modules*. The *pre-fetcher* is responsible for downloading video segments: this is done exploiting, at the same time, both the remote link (towards the video source) and the Wi-Fi Direct local mesh network (with the other video peers), which will be used by all the peers for sharing the segments downloaded from the video source. The segments downloaded are then queued in the *playout buffer*, which is drained by the *video player*.

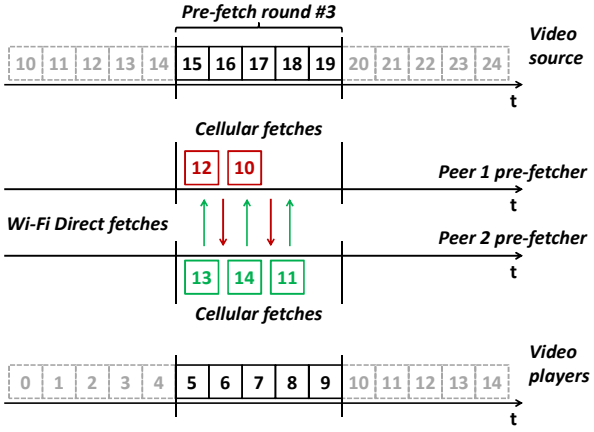

Fig. 3: example of time evolution of the video source and of the video peers

FIB of video peer 1	
Name prefix	**Output face**
---	---
…	…
ccnx:/prd	192.168.49.255:9695 (p2p0)
ccnx:/foo.eu	160.80.103.102:9695 (rmnet0)
ccnx:/foo.eu/video1_100/video1_11.m4s	192.168.49.2:9695 (p2p0)
…	…

FIB of video peer 2	
Name prefix	**Output face**
---	---
…	…
ccnx:/prd	192.168.49.255:9695 (p2p0)
ccnx:/foo.eu	160.80.103.102:9695 (rmnet0)
ccnx:/foo.eu/video1_100/video1_10.m4s	192.168.49.1:9695 (p2p0)
…	…

Fig. 4: FIBs of video players

Pre-fetcher logic

We introduce the following notation. Each video segment s has a length of T_s, expressed in seconds. A batch of P segments is called *window*, and a *pre-fetch round* is a series of fetches that begins after a full window has been published by the video source.

To avoid multiple remote downloads of the same segment, at the beginning of a pre-fetch round each video peer shuffles the sequence of the segments to be downloaded, and then tries to pull each segment s: if s is available on the Wi-Fi Direct mesh, the video peer will download it from the proximity network; if not, the peer will download it from its own preferred remote network and share it with the other peers.

This process is explained in Fig. 3, which depicts the case of two video peers downloading a window of $P=5$ segments. While the video source is publishing the segments 15÷19, as a result of the presence of the playout buffer each video peer is playing the segments 5÷9, while downloading the segments 10÷14 from both the remote and the local networks. To sum up, the total playout delay between the player and source is $2T_s$, because each playout buffer stores $2P$ segments.

Segment download

Each segment s can be downloaded in two ways: using a remote link towards the video source, or using the local Wi-Fi Direct network from another video peer. This mechanism is implemented using some rules in the FIB of each video peer.

Remote downloads are supported by pre-inserting a preloaded route (*remote route*) towards the server-prefix via the remote face. This remote route is visible in Fig. 4 in both peers, in which the entry 'ccnx:/foo.eu' points to the remote address 160.80.103.202 via the remote interface (rmnet0).

Wi-Fi Direct downloads are supported inserting a dedicated route (*proximity route*) towards a peer on the Wi-Fi direct network that has already downloaded the requested segment from its remote interface. These proximity routes are visible in Fig. 4 in both peers: video peer 1 has downloaded from the remote interface the segment 10, and thus video peer 2 will insert in its FIB a proximity route pointing to the video peer 1 via the Wi-Fi Direct interface (p2p0). It's worth noting that the proximity routes will be chosen instead of the remote route because of the longest prefix approach used in the FIBs. Each video peer can look for the desired segment on the Wi-Fi Direct mesh with the *proximity route discovery*. For each segment downloaded from the remote network, a video peer publishes on the Wi-Fi Direct mesh a signaling message called *Proximity Route Information* (PRI), containing all the data needed to establish a proximity route (the <IP,port> tuple and the estimated net remote rate of the peer, discussed later). This data unit follows the naming scheme we saw earlier:

PRI ccnx:/prd/video_BW/video_SN

Each FIB is preloaded with the entry 'ccnx:/prd' pointing to a preconfigured multicast address, used for searching the PRIs. Thus, if the desired PRI is found on the Wi-Fi Direct mesh, the peer will set up a route towards the peer that published the PRI. Note that the PRI publication and detection may not imply that the video peer has finished the corresponding remote download. Thus, each peer may become the splitting point of a multicast tree, and as soon as some chunks of a segment are received from the remote network, they are relayed both to the local pre-fetcher and to the Wi-Fi Direct interface.

Video coding rate selection

The selection of the video coding rate to be used in the next pre-fetch round is made at the end of each pre-fetch round and is identical to the one presented in [7]. The rate selection algorithm is based on i) all the available coding rates for the video, BW_h and ii) the *net* rate that each peer has obtained from the remote interface, C_i. Given L possible video coding rates, all the available BW_h values ($1 \le h \le L$) are discovered by each peer by parsing the MPD file, while the values $C_i(k)$ achieved in the pre-fetch round *k-1* are added by the node i to the PRI during the pre-fetch round k. This value is estimated considering the time needed to download the segments from the remote link.

Given M as the number of video peers in the Wi-Fi Direct mesh, the video coding rate selection relies on the resolution of the constrained maximization problem discussed in [7]:

$$J_{i,h} = \text{floor}\left[\frac{P\ C_i(k)}{BW_h}\right] \quad (1)$$

$$\max_h \left\{ s.t. \sum_{i=1}^{\min(P,M)} J_{i,h} \ge P \right\} \quad (2)$$

Equation 1 represents the number of complete video segments encoded at BW_h that a peer may download during a single pre-fetch round, while the maximization of (2) gives the highest possible video coding rate h whose segments can be completely downloaded within a round duration. The sum in (2) is limited to min(P,M) because at most P peers can carry out remote downloads. In addition, even if $P \ge M$, optimizing (2) may require excluding some peers from downloading segments from the remote link. This may happen when peers are obsolete or have poor coverage with respect to the others.

Of course, more comprehensive optimizations can be performed taking into account other factors like the costs of cellular data connections or the amount of remaining battery. In general we have seen that the GO (Group Owner) of the peer to peer network may become a bottleneck (see later), and may be wiser to exempt it from the group of peers remotely downloading the content.

IV. Android Implementation Issues

We implemented the application logic within the Android devices using the CCNx Android libraries. With respect to our previous Linux version [7], we had to cope with some Android limitations, which are hereafter described.

The download phase of a window impacts *heavily* on the heap of the mobile device, since every pre-fetch round means launching *2P* new background threads (P for the proximity route discovery and P for the real download). To avoid additional memory saturation, if a round should not end in the expected time (see next section), the following round will be launched only if the device has enough memory to handle both, and the rate of the round starting will be set to be the closest to the 60% of the coding rate of the "late running" round.

In addition, due to the used codec, we may end up with some video segments with different dimensions (Fig. 6) and some very large outliers may show up, saturating memory. We solved this problem storing the video segments on the flash NAND mass memory available on the device: its I/O speed is comparable with the bandwidth of the Wi-Fi, thus not limiting our solution, as the bottleneck is instead generally on the remote access network. Considering that real devices may show suboptimal unexpected performances, we ran a few tests and concluded that the current achieved cellular download rates are almost one order of magnitude slower than the I/O speed of the device. The tests were made using a HTC One S smartphone, and gave a result of about 20 MB/s, higher than the rates we obtained on the cellular network (see next section).

The choice of using Wi-Fi Direct comes from another Android limitation: indeed, Android do not allow using the cellular network and the Wi-Fi infrastructure ports in parallel, while it is permitted using the Wi-Fi Direct port in parallel to the cellular or the Wi-Fi infrastructure port.

V. Tests and Results

The testbed we used is a set of real different Android 4.0+ cellular devices (smartphone and tablets), running CCNx 0.8.1 and VLC 0.1.0. We used as reference video the movie "Big Buck Bunny", composed by about 270 480p segments long T_s = 2 sec each and encoded at fourteen rates, ranging from 100 kbps to 4.5 Mbps. We point out that Android devices can use any data connection (cellular or Wi-Fi) together with Wi-Fi Direct: the exploitation of multi radio access technology environment is one of the great benefits of this approach. Some tests have been carried out on multi-RAT but they are not reported for space reason. To show instead the effects of cooperation we chose a single Radio Access Technology for all the devices (WLAN or WWAN) in which the results could be analyzed without influences of other external factors.

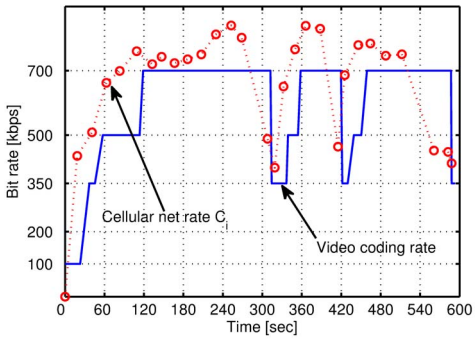

Fig. 5: video coding rate achieved with a single device (without cooperation)

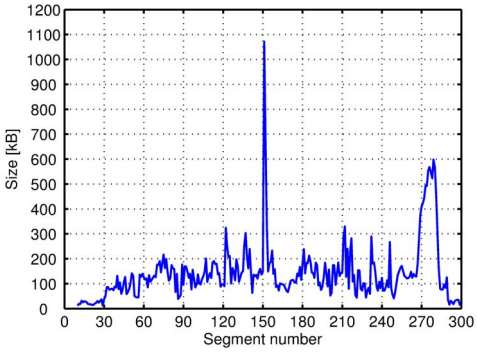

Fig. 6: size of segments downloaded during the test of Fig. 5

The first test aims at finding out the performance of our application on a single device, thus not considering the cooperation among all the neighboring devices. We used a Samsung Galaxy Note 2, connected to a 3G network with a pre-fetch window set to *P=10* segments. The results are shown in Fig. 5: the maximum video coding rate of 700 kbps obtained in the test reflects the real downstream rate achieved by the device, which never goes beyond 900 kbps. The cause of the bit rate drop around second 300 can be found in Fig. 6, in which we report the size of the downloaded segments; the encoder used for the movie gave some outliers, independent from our solution and not predictable by the coding rate selection algorithm. Nevertheless, our application can react to this situation, decreasing the quality of the video stream.

The second test aims at evaluating the usefulness of the cooperation between two nodes, in terms of video coding rate enhancements. We repeated the previous test, using a Samsung Galaxy Note 2 and a HTC One X collaborating from the beginning of the video stream. The pre-fetch window was set to *P=10* segments. The results are shown in Fig. 7. In the first pre-fetch round each device estimates its downstream rate downloading the segments only from the cellular interface.

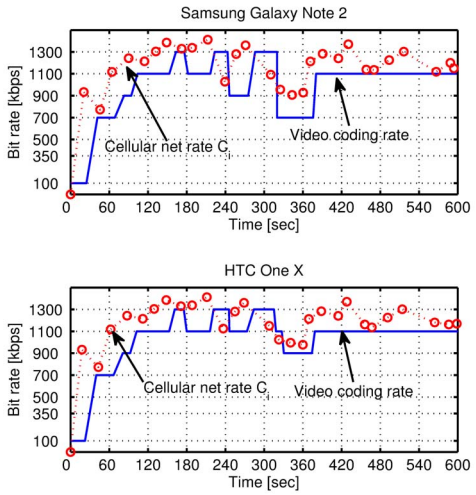

Fig. 7: video coding rate achieved with two devices

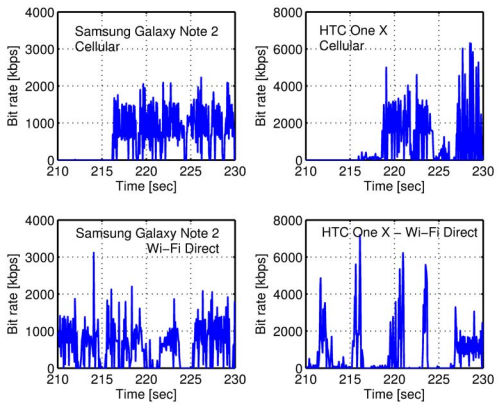

Fig. 8: activity on interfaces during a given time period

From the second pre-fetch round the collaboration takes place, enhancing the coding rate to 1100 kbps on each device: this behavior is due to the activity on the Wi-Fi Direct mesh, which allows each peer to exploit the cellular resources to download segments at a higher bit rate in the same pre fetch round. Note that the small differences of behavior of the two devices are due to the current status of the device itself (CPU, installed, running apps, and so on). As proof of the activity on both channels we provide, in Fig. 8, a snapshot of the activities on both devices in the pre-fetch between second 210 and 230: both devices are exploiting at the same time both the cellular and the Wi-Fi Direct interface.

The third test is related to the pre-fetch window. As underlined in [7] this is a critical value, because it is directly related with the delay that the video player suffers before it can start the playback. In addition, larger pre-fetch window can lead to a great increase of resource consumption on the device. We ran a series of tests with the same two devices, starting from a pre-fetch window of *P=10* segments and lowering it down to *P=2* segments. The results are shown in Fig. 9 and in Fig. 10, from which we can draw the following conclusions:

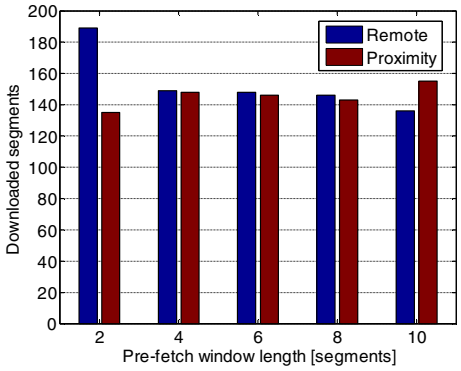

Fig. 9: fairness in segments download

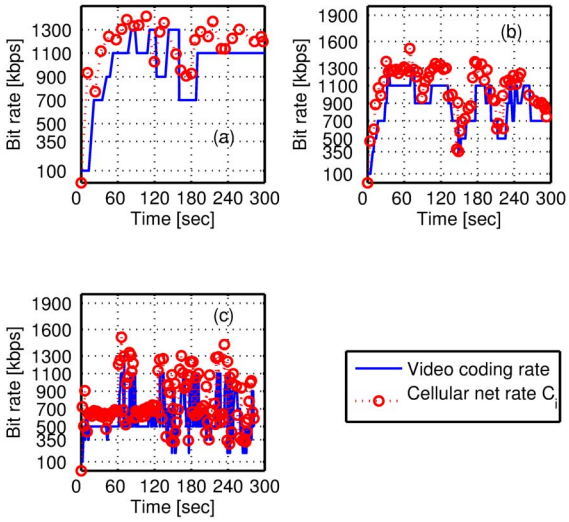

Fig. 10: variation of aggregate bandwidth and video coding rate upon pre-fetch window decrease

- For windows of length $P \geq 3M$ (Fig. 10a, $P=10$) the average video coding rate is about 1050 kbps, and the collaboration between the nodes is fair (Fig. 9). In this configuration we have the best stability of the coding rate, because of the floor operator of equation (2).
- For windows of length $P=2M$ (Fig. 10b, $P=4$), the coding rate becomes lower (about 850 kbps), and the video quality becomes unstable. Nevertheless, we still have fairness in the remote downloads (Fig. 9).
- For windows of length $P<2M$ (Fig. 10c, $P=2$) the video coding rate decreases again, as well as the stability of its selection. In addition, we can see from Fig. 9 that there is no more fairness in the remote downloads.

In the last test we analyzed how traffic overloads impact on the Wi-Fi Direct group owner (GO). We ran the test increasing the number of nodes and varying the pre-fetch window as underlined previously. Up to four nodes we reported no issues. In Fig. 11 we report the duration of each round achieved by the group owner in the case of five Samsung Galaxy Note 2 with pre-fetch window set to $P=20$. After the first four rounds, the group owner cannot keep up with the traffic generated in the Wi-Fi Direct link: Wi-Fi Direct networks have a star topology with the GO at its center, so the GO is responsible for the forwarding of *all* the traffic in the mesh. Due to the dependencies on the computational capabilities of the devices used, the lesson learnt is to choose a GO among the more powerful devices, as this is a point of failure in the system.

We point out that the physical layer of Wi-Fi Direct is the same of the Wi-Fi Infrastructure. What change in Wi-Fi Direct are the computational and network loads that the GO suffers, since does the job of relaying every message. In addition, the saturation of the GO comes without any noticeable delay in data processing. A powerful GO may be able to support a larger number of active participants to the group. In addition, it's worth noting that if we consider having a good bit rate with modern cellular networks, small groups of peers may be enough to achieve a good video quality on a high bit rate streaming download.

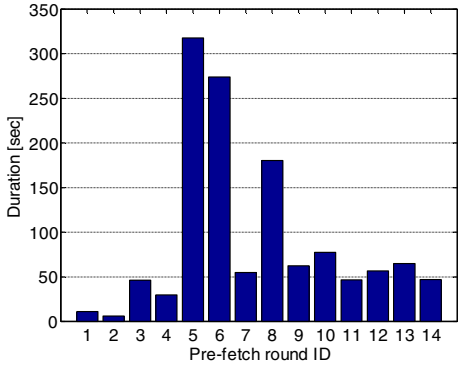

Fig. 11: pre-fetch rounds durations in the group owner in the case of five nodes

VI. CONCLUSIONS

We developed an Android application for the live streaming of MPEG-DASH encoded videos, exploiting the functionalities of ICN and Wi-Fi Direct to allow a better user experience on different commercial mass market Android devices. We have demonstrated the feasibility of this approach, and gave a sketch of the bottlenecks that may hinder or limit the benefits of collaboration.

Two main conclusions can be drawn. First: it is possible to organize a group of commercial devices to cooperate in a streaming activity and reduce the amount of transmitted data to a minimum. Second: the bottleneck of the system of the Group Owner, due to the star configuration of the Wi-Fi Direct group.

Although this topic is known, to the best of our knowledge we are the first to exploit such a solution on real common hardware, without modifying the underlining behavior of the operating system. We strongly believe that this is actually the scenario in which the benefits are higher and the analyses we report represent the missing link to close the gap to real implementations.

REFERENCES

[1] Van Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard, "Networking Named Content," in Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Rome, Italy, 2009.

[2] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, M. Varvello, From content delivery today to information centric networking, Computer Networks, Volume 57, Issue 16, 13 November 2013, Pages 3116-3127.

[3] T. Stockhammer, "Dynamic adaptive streaming over HTTP: standards and design principles", ACM MMSys 2011

[4] T. Koponen, M. Chawla, B.G. Chun, et al. "A data-oriented (and beyond) network architecture", ACM SIGCOMM 2007

[5] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, "Supporting the Web with an Information Centric Network that Routes by Name", Elsevier Computer Networks, vol. 56, Issue 17, p. 3705–3722.

[6] Wi-Fi Alliance "Wi-Fi Direct". http://www.wi-fi.org/discover-wi-fi/wi-fi-direct

[7] Andrea Detti, Bruno Ricci, and Nicola Blefari-Melazzi. "Peer-to-peer live adaptive video streaming for Information Centric cellular networks." Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on. IEEE, 2013.

[8] A.Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, A. Bragagnini, "Offloading cellular networks with Information-Centric Networking: the case of video streaming", IEEE WoWMoM 2012

[9] Derek Kulinski and Jeff Burke, "NDN Video: Live and Prerecorded Streaming over NDN", NDN Technical Report NDN-0007

[10] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, A. Markopoulou, "Microcast: Cooperative Video Streaming on Smartphones", ACM MobiSys 2012

[11] Z. Xiaoqing, P. Agrawal, J.P. Singh, et. al "Distributed Rate Allocation Policies for Multihomed Video Streaming Over Heterogeneous Access Networks" IEEE Transaction on Multimedia, Vol. 11, No. 4, June 2009

[12] B. Han, N. Choi, T. Kwon, Y. Choi, "AMVS-NDN: Adaptive Mobile Video Streaming and Sharing in Wireless Named Data Networking" , IEEE NOMEN 2013

[13] J. J. D. Mol, A. Bakker, J. A. Pouwelse, D. H. J. Epema, H. J. Sips, "The Design and Deployment of a BitTorrent Live Video Streaming Solution," IEEE International Symposium on Multimedia, 2009

[14] Cisco "The Zetta-Byte era: trends and analysis", June 2014

[15] RFC 6824 "TCP Extensions for Multipath Operation with Multiple Addresses" Jan 2013

[16] Y. Kojima, J. Suga, T. Kawasaki, M. Okuta and R. Takechi. "LTE-WiFi Link Aggregation at Femtocell Base Station". World Telecommunication Congress 2014. Berlin, Germany.